Implementation in Workstation Environment

APPS_DEFAULTS

Purpose

- Designed to eliminate direct references to site specific information from programs and scripts.Resolves site specific information without direct references in the code.
- Advantages
  - Allows different sites to configure directories and parameters as needed.
  - Allows different users at a site to override system-wide settings for test purposes.
  - Allows any site or user to add to the existing set of apps defaults

APPS_DEFAULTS

Definition of Terms

- Token – a code word naming specific information used in a program or script.
- Value – the site specific information.
- Format – token : value
- Nested tokens allow for tokens to be part of a value
  - Use $(token) format
- Examples:
  - rfs_dir : /awips/hydroapps/rfc/nwsrfs
  - ofs_dir : $(rfs_dir)/ofs
  - db_name : hd5_11ofstest

APPS_DEFAULTS

Storing Pairs

- Token : Value pairs are stored in files or environmental variables.
- Hidden files located by environment variables that are defined when you login.

File	Environment Variable	Usual Location
User	APPS_DEFAULTS_USER	$HOME/.Apps_defaults
Site	APPS_DEFAULTS_SITE	/awips/hydroapps/.Apps_defaults_site
National	APPS_DEFAULTS	/awips/hydroapps/.Apps_defaults

APPS_DEFAULTS

How it works

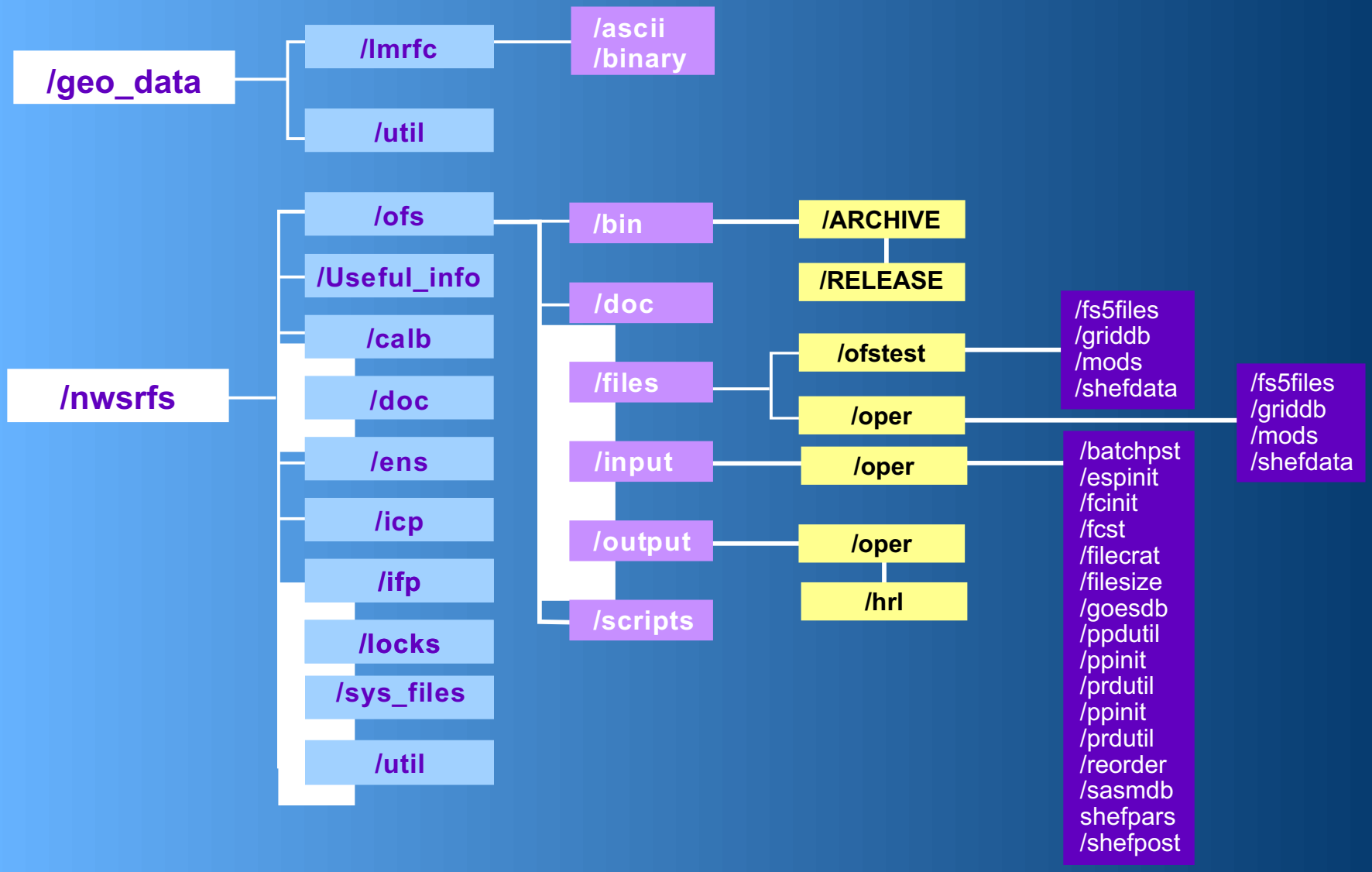
- Token information is resolved by requesting the value of a token using the get_apps_defaults function/program within a program or script.
- Apps_defaults resolves the value of the tokens by looking in the following places, in order:
  - Environment variables,
  - User file,
  - Site file,
  - National file.

APPS_DEFAULTS

Useful Scripts and Functions

- gad
  - Symbolic link to get_apps_defaults - easier to type
  - Usage: gad token_name
- gad_w
  - Script to find the location of the token value in effect
  - Usage: gad_w token_name
- go
  - Uses apps_defaults to help navigate thru directory structure
  - Need to know some basic token names
  - Function set up by the fun script
  - Usage: go token_name
  - Example: go rfs_dir
    - Result: cd /awips/hydroapps/rfc/nwsrfs

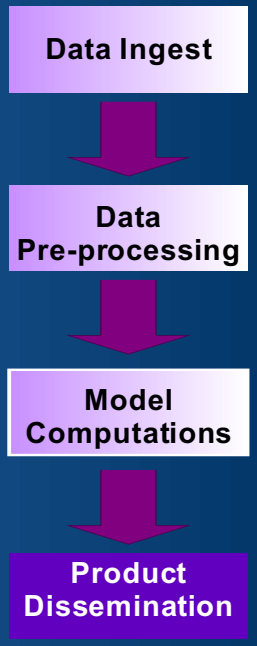
OFS Directory Structure



OFS Script

Purpose

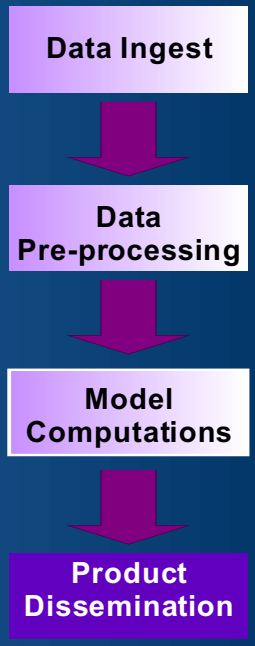
- The ofs script resides in the $(ofs_scripts) directory.
- It is the front end for running all ofs programs.
- Most offices have created their own front ends that sit on top of the ofs script.



OFS Script

Running the script

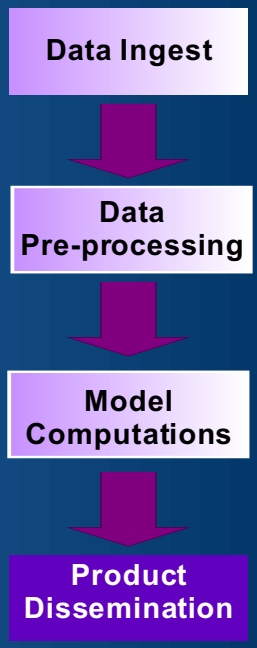
- The command string
ofs -p *program_name* -i *input_file* -o *output_file*
- Executes program in $(ofs_rls) directory
- Reads input from $(ofs_input)/*program_name*
- Writes output to $(my_output)
- Types of output:
  - Print
  - Punch
  - Log file



OFS Locks

Purpose

- In order to keep multiple processes from colliding, each process requests a lock on the database before it can begin.
- Types of locks
  - Read
  - Write
- Features
  - Multiple read locks permitted.
  - Only one lock (either read or write) permitted when write lock requested.



OFS Locks

Waiting for a Lock

- Each process checks every few seconds to see if it can get a lock. Wait length, ofs_lock_wait_interval apps_default.
- No queuing. The first process you send in may not be the first to get a lock.
- Checks for some period of time. Total wait time, ofs_lock_max_wait apps_default.

OFS Locks

Chart of Actions

Case	Request	File Open?	Current Lock	Granted?
1	R	N	NA	Y
2	R	Y	R	Y
3	R	Y	W	N
4	W	N	NA	Y
5	W	Y	R	N
6	W	Y	W	N